

A Fast and Robust 3D Feature Extraction Algorithm for Structured Environment Reconstruction

Jan Weingarten¹, Gabriel Gruener², Roland Siegwart¹

¹Swiss Federal Institute of Technology Lausanne (EPFL)
Autonomous Systems Lab
CH-1015 Lausanne, Switzerland
jan.weingarten@epfl.ch

²Centre Suisse d'Electronique et de Microtechnique (CSEM)
Microrobotics Division
CH-6055 Alpnach Dorf, Switzerland
gabriel.gruener@csem.ch

Abstract

This paper describes an algorithm for generating compact feature-based 3D models of indoor environments with a mobile robot. The emphasis lies on the high performance of the algorithm, its possible incremental use, as well as its wide applicability to a variety of sensors as it does not assume any structure in the raw data at all. It recovers planar surfaces of physical environments based on a set of unorganized points $\{v_1, \dots, v_N\} \in \mathbb{R}^3$ and generates a compact, real-time renderable 3D model.

1 Introduction

Since standard computers allow efficient processing and visualization of three-dimensional data, the interest of using 3D graphics has increased in numerous fields. Car design, architecture, or the modern movie industry are merely a few examples of fields which are unimaginable nowadays without the use of 3D graphics. Whereas the visualization hardware is powerful, the process of developing 3D models is still a time-consuming task. To make this process more efficient, range sensors can help. In commercially available stationary 3D scanning systems for example, range sensors capture objects from different perspectives and a specialized software transforms the obtained 3D data into a three-dimensional model.

However, in other areas an autonomous mobile robot with a 3D scanning system could be more useful. In urban search and rescue applications or generally in terrain inaccessible or hazardous for human beings, a robot generating 3D models of physical environments could provide enormous help for planning operations. In addition, architects or building managers could use environmental 3D models for design and utility studies. The video game industry could also benefit from such automatic 3D-modeling systems. And finally, the robot itself could improve its navigation reliability using a three-dimensional map, as its dense information may reduce the position estimation ambiguities commonly met in 2D navigation systems.

Compared to stationary 3D scanning systems with a precisely known position within a deviation of some millimeters, the position uncertainty of a moving robot varies over time depending on the quality of the sensory information. Erroneous odometric data and a reduced field of view caused for example by persons standing around the robot can limit the localization accuracy of a mobile robot to a range from a few centimeters to some decimeters.

Additionally, the generated data is subject to noise due to limitations of the measurement process of the sensor not adapted to certain materials like glass and can contain irregularities like holes (doorways). Furthermore, the data is really three dimensional and not only 2.5 dimensional, like in stationary laser scanning systems. Thus, next to efficiently processing the high amount of data – a common problem of 3D measurement applications – the algorithm adapted to a moving 3D-scanning system has to cope with noisy and irregular data.

To reduce the complexity of the robotic 3D mapping problem, it is assumed that in structured environments like office corridors or urban areas, the majority of occurring objects are walls, ceilings or building facades that can be represented by simple geometric primitives like planes.

1.1 Motivation

Most existing robotic 3D mapping approaches are based on a 2D localization system using a horizontal laser scanner and an additional vertical laser scanner for 3D scanning (see [9], [7], [3]). They exploit the fact that the robot generates consecutive scans as it moves through the environment. It is therefore assumed that the chronological order of the laser scans corresponds to the spatial order meaning that different scans are spatially separated. This allows to connect consecutive scan data resulting in nice looking but quite complex 3D models. However, if the robot moves around a corner and rotates, overlapping scan data may produce unsatisfactory results, as a simple connection of consecutive scan points would lead to

wrong surfaces. Besides, if the used sensor would not be a laser scanner like the SICK LMS for example, but a stereo camera or a time-of-flight camera, consecutive scans could no longer be easily connected. The proposed algorithm therefore does not make any assumption about the topology of the data. It takes unorganized three-dimensional data points as input and outputs all rectangular planar regions approximating the measured physical environment.

2 Previous Work

In general the field of 3D mapping is not very vast. The approaches can be categorized into those that assume knowledge of the robot and those that do not, which can be called 3D SLAM (*simultaneous localization and mapping*).

The majority of approaches use a vertical laser scanner that creates registered 3D data by sweeping through the environment. Thrun et al. [9] describe an approach to real-time SLAM with applications to 3D mapping. Their algorithm connects consecutive scans and reduces the complexity of the model by standard polygonal simplification methods from the field of computer graphics. Haehnel et al. [3] describe an algorithm for indoor and outdoor environments that generates less complex 3D models by extracting planar features with a region-growing technique. The computation time needed is several minutes. Liu et al. [7] use a similar setup as Haehnel but extract planes with the expectation maximization (EM) paradigm and maps textures onto the planar features found. This results in nice looking 3D models but takes several minutes of computation time and is only applicable to complete datasets and therefore can't be used for incremental map building.

Iocchi et al. [5] also generate texturized 3D maps but use a stereo system to scan the environment. Range information as well as intensity information is used to estimate camera motion. This approach thereby falls into the category of approaches that do not assume precise sensor location but require some manual guidance in the reconstruction process.

Feddema and Little [1] describe how to extract single planes and other parametric objects of manually segmented data by linear regression. The algorithm we proposed in this paper uses the same plane extraction procedure but performs the segmentation automatically. It does not require any manual intervention.

Another field of interest is surface reconstruction within computer graphics.

Hoppe et al. [4] describe how to reconstruct arbitrary surfaces from unorganized points. They decompose the point cloud into smaller neighborhoods, fit tangent planes to each neighborhood, and reconstruct a closed surface using the marching cubes algorithm.

Roth and Wibowo [6] also reconstruct surfaces from range data with the marching cubes algorithm and use a similar decomposition scheme of the space as the one used in this work. But in this work, the data structure used by Roth and Wibowo is replaced by a dynamic list.

In fact, if the time stamp information of the different scans is not used for surface reconstruction purposes, the robotic 3D mapping problem reduces to a standard surface reconstruction problem. The only difference lies in the noisiness of the data and the irregularities as doorways or windows may produce severe holes in the dataset.

3 Description of the Algorithm

The divide-and-conquer strategy generally consists in breaking a problem into simpler subproblems of the same type, solving these subproblems, and finally amalgamating the obtained results into a solution to the overall problem.

In this case the input data set is divided into small cubic neighborhoods (*dividing step*). These neighborhoods are then approximated by best-fitting planes (*solving the subproblems*), and merged to form the final surface model (*merging step*).

The division into small cube cells is described in section 3.1. Section 3.2 describes the tangent plane approximation and section 3.3 the region growing method.

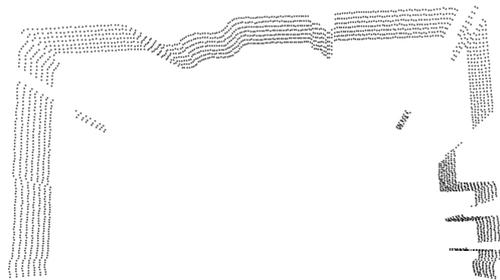


Figure 1: Seven raw registered SICK LMS scans (2527 vertices) forming a simple raw data set. It represents a part of a corridor at EPFL with walls on the left and the right hand side and a ceiling in the top.

3.1 Discretization of the Point Cloud

The input raw data is an unorganized point cloud composed of thousands of 3D data points $\{v_1, \dots, v_N\} \in \mathbb{R}^3$ representing structured environments, like office corridors for example, made up by a number of walls, a ceiling, doors, and other objects. It could as well represent urban outdoor environments

consisting of planar structures like building facades or walls.

In order to be able to extract all relevant planes from the raw data, a segmentation method has to be found allowing to distinguish between data relevant to a certain plane fit and data that is not. Furthermore, to achieve high efficiency, the segmentation time as well as the plane extraction time should be kept as low as possible.

A decomposition of the 3D space into an array of regular cubes containing each the spatially corresponding data points fulfills these requirements. Firstly, the data points included in a cube are much more likely to belong to one single plane. Secondly, as the number of vertices contained in a cube is a small fraction of the overall number of vertices depending on the adjustable cube cell size, the plane extraction will be very fast.

A disadvantage of decomposing the space into small cubes is certainly the extra amount of memory required. Octree structures, dynamic or run-length encoded lists provide efficient ways of overcoming this drawback. In this work, a dynamic list structure has proven to be the most suitable as it enables faster sequential traversal of all cube cells than octrees and needs the least amount of memory.

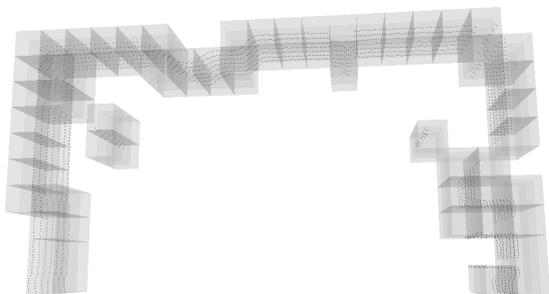


Figure 2: The result of the decomposition of the data shown in Figure 1 into 156 small cube cells. In this case, each cube cell has a side length of 30 centimeters and contains between 1 and 51 vertices.

3.2 Finding the Tangent Planes

This section describes how the algorithm approximates the content of one cube cell by a rectangular plane patch which can be called a *quad*, a rectangular plane with a length and a width. After having decomposed the space into k small cube cells c_j and assigned every raw data point v_i to its corresponding cell, the list of cells is sequentially run through and a quad q_j is fit to every cell c_j by the following steps:

3.2.1 Segmentation

The RANSAC segmentation paradigm presented by Fischler and Bolles [2] helps to separate the points in a cell into points that belong to a plane and points that do not. Without this segmentation step, it would not be assured that found planes actually belong to physical entities, as direct least-square fitted models are not necessarily close to a large number of supporting vertices.

3.2.2 Least-Square Plane Fitting

The fitting step takes as input the output of the segmentation step which is an amount of points belonging to a plane and performs a least-square fit by the method described below.

A plane in \mathbb{R}^3 is defined by $\mathbf{u} \cdot \mathbf{p} - d = 0$ (Hesse notation) where $\mathbf{u} = (u_x, u_y, u_z)^T$ is a unitary vector perpendicular to the plane, $\mathbf{p} = (p_x, p_y, p_z)^T$ is a 3D position vector and d is the perpendicular distance from the plane to the origin. This formula is identical to $u_x x + u_y y + u_z z - d = 0$ and can be simplified to $n_x x + n_y y + n_z z + 1 = 0$ with $d \neq 0$, allowing to find a closed form solution for the regression problem formulated below. The constraint $d \neq 0$ means that the perpendicular distance from the origin to the plane must not be zero. This can easily be achieved by defining the world coordinate frame in an empty space ensuring no sensor data will occur near the origin. The sum of least-square distances which has to be minimized is defined by

$$S = \sum_{i=1}^N (n_x x_i + n_y y_i + n_z z_i + 1)^2$$

To find the minimum error, this sum has to be partially derived with respect to n_x , n_y , n_z and set to 0. The result in matrix notation then reduces to

$$\mathbf{n} = \mathbf{A}^{-1} \cdot \mathbf{b}$$

with

$$\mathbf{A} = \begin{pmatrix} \sum x_i^2 & \sum x_i y_i & \sum x_i z_i \\ \sum x_i y_i & \sum y_i^2 & \sum y_i z_i \\ \sum x_i z_i & \sum y_i z_i & \sum z_i^2 \end{pmatrix}$$

$$\mathbf{b} = \begin{pmatrix} -\sum x_i \\ -\sum y_i \\ -\sum z_i \end{pmatrix}$$

$$\mathbf{n} = \begin{pmatrix} n_x \\ n_y \\ n_z \end{pmatrix}$$

The distance d of the found plane to the origin is given by

$$d = \frac{1}{\sqrt{n_x^2 + n_y^2 + n_z^2}}$$

and the components of the plane normal vector $\mathbf{u} = (u_x, u_y, u_z)^T$ are $u_x = -dn_x$, $u_y = -dn_y$ and $u_z = -dn_z$ (see Feddema and Little [1]).

Alternatively, least-square plane fitting can be done by principal component analysis of matrix \mathbf{A} as described by Hoppe et al. [4] or other methods as described in Wang et al. [10]. These methods have the disadvantage that the parameters have to be calculated numerically whereas the method presented in this work uses a closed form solution.

Besides least-square fitting methods, voting schemes like the 3D Hough Transform can be used to extract planes reliably as described by Iocchi et al. [5]. But these tend to be slower in practice and less precise as they require discrete data.

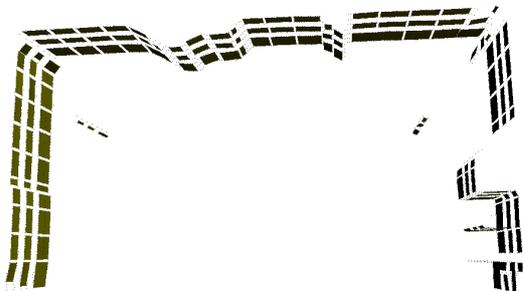


Figure 3: In every cube cell containing a minimum number of 6 vertices, a tangent plane is fitted and adjusted to the data, forming a quad. In this case, the 156 input cubes are approximated by 127 quads. The residual 29 cube cells contain less than 6 vertices and are therefore not included in the reconstruction process.

3.2.3 Sizing

After having determined the parameters of the (indefinite) plane, the extent of the plane is evaluated. Firstly, all points of the cube cell supporting the plane are projected onto the plane. Then, these projected plane points are rotated and translated into the global x, y -plane defined by $z = 0$. Finally, a two-dimensional bounding box is calculated which is rotated and translated back into the original plane location. The formerly indefinitely large plane is thereby reduced to a quad.

The result of this step is a list of separated quads $Q = \{q_1, q_2, \dots, q_k\}$ approximating the physical environment. A quad q_j is defined by 3 vertices v_1, v_2, v_3 defining two spanning vectors $\mathbf{a}_j = \overline{v_1 v_2}$ and $\mathbf{b}_j = \overline{v_1 v_3}$, a normal \mathbf{n}_{q_j} and a center of gravity \mathbf{cog}_{q_j} defined as the average value of the supporting vertices of the plane within the cube cell c_j .

3.3 Region Growing

After having obtained a best fitting plane for every cube cell (see Figure 3), neighboring quads q_i, q_j are merged under certain criteria to form bigger quads. Two constrains define these merging criteria:

1. Matching orientation:

$$|\mathbf{n}_{q_i} \cdot \mathbf{n}_{q_j}| > a_{min}$$

evaluates whether the orientation of quad q_i matches the orientation of quad q_j up to a threshold value a_{min} .

2. Matching translation:

$$|\mathbf{n}_{q_i} \cdot \mathbf{cog}_{q_j} - \mathbf{n}_{q_i} \cdot \mathbf{cog}_{q_i}| < d_{max}$$

evaluates whether the center of gravity cog_{q_j} of quad q_j lies near the plane of quad q_i up to a threshold of d_{max} .

If these two constrains are satisfied, the algorithm merges the two quads. To merge quads in an efficient way, the algorithm benefits from the topological order of the cube cells. Hence, every quad has to be compared in the worst case to thirteen neighboring quads saved in proceeding positions of the quad list. The final model consists of the residual number of quads which represent rectangular areas of the input data (see Figure 4).

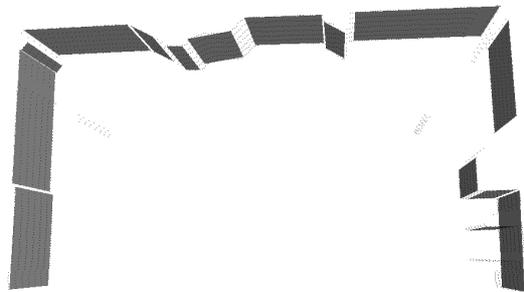


Figure 4: The resulting final model consists of 17 planar regions. It can be seen that the chosen level of detail (cellsize of 30 centimeters) allows to recover more than just the two main walls and the ceiling. Almost every planar structure is detected, like the book shelves on the bottom right.

The algorithm can be fine tuned by adjusting parameters like a_{min} , d_{max} , the cube cell size, and the minimum number of vertices which have to be included in one cube cell in order to represent a quad. Especially the cube cell size can influence the outcome of the algorithm drastically as it defines the resolution and the related number and size of the cube cells.

3.4 Complexity Analysis

The algorithm presented in the last section is divided into three steps. In the first step, the space is decomposed into k cube cells and the overall n vertices are distributed within the corresponding cube. As the k cube cells are sorted which has a complexity of $O(k \log k)$ (*quick-sort*), the overall complexity of the first step is $O(n + k \log k)$. The second step consists firstly of a segmentation step with the RANSAC algorithm which has a complexity of $O(k n_j^2)$. This can be reduced to $O(k)$ if the number of vertices n_j in one cell is regarded as constant. Secondly a tangent plane is fit to the n_j vertices of cube cell c_j by linear regression and additionally evaluating the parameters of the quad. This is done in $O(n)$. The third and last step performs a region growing algorithm on the cube cell level which can be done in $O(n)$ as the cube cell list is sorted.

creation of the data structure	$O(n + k \log k)$
plane fitting	$O(n)$
quad sizing	$O(n)$
region growing	$O(n)$
total algorithm	$O(n + k \log k)$

If the algorithm is used in an incremental way, the creation of the data structure will be possible in $O(c_{new}(\log k))$ where c_{new} denotes the number of newly scanned points gathered as the robot moves. This shows that if the algorithm is implemented in an incremental way and the number of new scan points c_{new} is constant, the creation of the data structure will be possible in $O(\log k)$. The steps following data structure creation are all of linear complexity as can be seen in the experimental verification in Figure 5.

4 Results

It is not easy to compare previously existing approaches to this one. First of all, the goal of this work is to extract the most possible planar regions of an unorganized point set. It doesn't aim to reconstruct a closed surface which is for example done in the approach of Hoppe et al. [4]. Furthermore, it does not assume topological order in the raw data which is generally done in all previously mentioned approaches using mobile robots and laser scanners ([7], [3] and [9]). Therefore, initial plane estimates cannot be found by connecting consecutive scan points, but have to be found by linear regression. Finally, rapidly evolving PC hardware also makes the comparison to existing methods more difficult.

4.1 Hardware Setup

The hardware setup used is similar to the setup proposed by Thrun et al. [9]. The robot has already

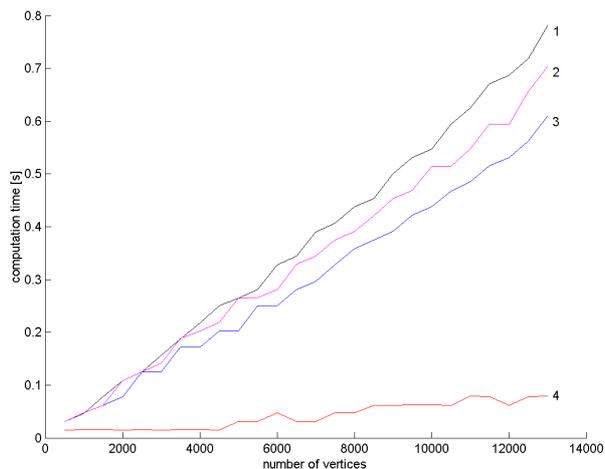


Figure 5: This graph shows the complexity of the different steps of the algorithm. The x-axis represents the number of vertices and the y-axis the necessary computation time. Curve 1 shows the computation time necessary for all processing steps in relation to the number of input vertices. Curve 2 shows the complexity of the quad fitting step together with the region growing step which are also represented separately by curve 3 (quad fitting) and curve 4 (region growing).

a 2D localization system based on two horizontally oriented SICK LMS laser scanners scanning a field of view of 360 degrees. The localization system was developed by Arras et al. [11] and uses linear features to localize the robot with sub-centimeter precision. An additional vertically mounted SICK laser scanner captures the surrounding environment profile while the robot moves through the environment. Registering these two dimensional slices of 3D information with 2D position estimates provided by the existing localization system of the robot yields a cloud of registered 3D data points.

4.2 Experimental Results

As an example, a piece of a corridor of a length of 7.5 meters was traversed. 37 scans were taken with a SICK LMS sensor generating in total 13357 3D points. The algorithm took an overall time of 0.75 s to recover 27 regions (see Figure 6). It decomposed the space into 566 cube cells containing between 1 and 90 vertices. The number of iterations for the RANSAC algorithm was set to 100.

A standard Pentium IV with 1.8 GHz was used for this analysis. The software was written under Windows 2000 in C++ and VTK (*The Visualization Toolkit*), which is a freely available data processing and visualization library.

5 Conclusion and Future Work

In this work an algorithm for finding planar regions in a set of unorganized 3D points is presented. Compared to existing approaches, it is efficient and does not assume an existing topology in the raw data, making it usable for any 3D sensor available.

Furthermore it produces very compact 3D models and is designed to be extensible to processing and propagating uncertainty information which will be the next research steps. Another future development will be the incremental implementation and intense testing with different sensor types like a time-of-flight camera. As not all planar surfaces are suited to be approximated by quads, planar polygons could be used instead. Finally, the last step will consist of mapping textures onto the extracted 3D features.

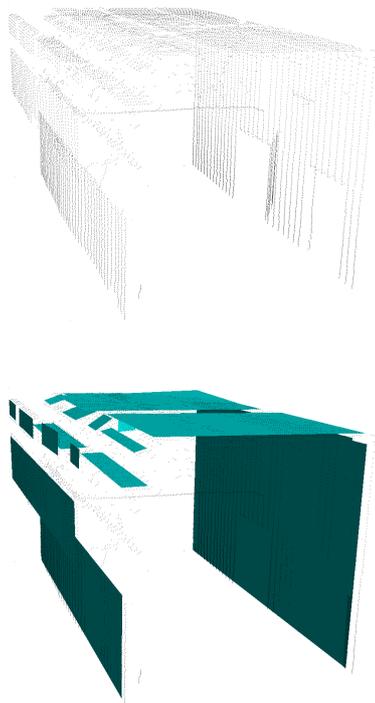


Figure 6: Results for a more complex scene. The upper image shows the raw data consisting of 13157 vertices representing a typical corridor of an office environment. The image below shows the resulting 3D model built of 27 planar regions. Note that the two doorways are not reconstructed.

References

- [1] John T. Feddema, Charles Q. Little, "Rapid World Modeling: Fitting Range Data to Geometric Primitives", *Proceedings of IEEE ICRA*, 1997, Volume 4, pp. 2807-2812.
- [2] Martin A. Fischler and Robert C. Bolles, "Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography", *Communications of the ACM*, June 1981, Number 6, Volume 24, pp. 381-395.
- [3] Dirk Hähnel, Wolfram Burgard, Sebastian Thrun, "Learning Compact 3D Models of Indoor and Outdoor Environments with a Mobile Robot", *The fourth European workshop on advanced mobile robots (EUROBOT'01)*, 2001, pp. 91-98.
- [4] Hugues Hoppe, Tony DeRose, Tom Duchamp, John McDonald, Werner Stuetzle, "Surface Reconstruction from Unorganized Points", *ACM SIGGRAPH 1992*, July 1992, pp. 71-78.
- [5] Luca Iocchi, Kurt Konolige, Max Bajracharya, "Visually Realistic Mapping of a Planar Environment with Stereo", *Proc. of Seventh International Symposium on Experimental Robotics (ISER)*, Hawaii, 2000, pp. 521-532.
- [6] Gerhard Roth, Eko Wibowo, "A Fast Algorithm for Making Mesh-Models from Multiple-View Range Data", *Proceedings of the Robotics and Knowledge Based Systems Workshop*, 1995, pp. 349-355.
- [7] Yufeng Liu, Rosemary Emery, Deepayan Chakrabarti, Wolfram Burgard, Sebastian Thrun, "Using EM to Learn 3D Models of Indoor Environments with Mobile Robots", *Proceedings of the IEEE International Conference on Machine Learning (ICML)*, 2001, June 28 - July 1, 2001, pp. 329-336.
- [8] Sebastian Thrun, "Robotic Mapping: A Survey", *Exploring Artificial Intelligence in the New Millennium*, Morgan Kaufmann, February 2002.
- [9] Sebastian Thrun, Wolfram Burgard, Dieter Fox, "A Real-Time Algorithm for Mobile Robot Mapping With Applications to Multi-Robot and 3D Mapping", *ICRA*, San Francisco, April 2000, Volume 1, pp. 321-328.
- [10] Caihua Wang, Hideki Tanahashi, Hidekazu Hirayasu, Yoshinori Niwa, Kazuhiko Yamamoto, "A Probabilistic Approach to Plane Extraction and Polyhedral Approximation of Range Data", *IEICE Trans. Inf. and Syst.*, February 2002, Number 2, Volume E85-D, pp. 402-410.
- [11] K.O. Arras, N. Tomatis, B. Jensen, R. Siegwart, "Multisensor On-the-Fly Localization: Precision and Reliability for Applications", *Elsevier, Robotics and Autonomous Systems*, 2001, 34 (2-3), 131-143.